National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cybersecurity

Chetanya Sharma
Student ID: x23422793

School of Computing
National College of Ireland

Supervisor: Dr. Arghir-Nicolae Moldovan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	……Chetanya Sharma……
Student ID:	x23422793
Programme:	MSc. in Cybersecurity …… **Year:** …2025……..
Module:	MSc (Research) Practicum/Internship Part 2
Lecturer:	Arghir-Nicolae Moldivan
Submission Due Date:	11th December,2025
Project Title:	A Comparative Analysis of OWASP ZAP, Arachni and Wapiti Against the Reinforced WAVSEP Benchmark
Word Count:	1500 **Page Count:** 10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:

Date: 9th December, 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Chetanya Sharma
Student ID: x23422793

1 Environment Setup

This section helps in preparing the host system, virtual machine and the base environment that are required to reproduce the DAST benchmarking setup using the Reinforced WAVSEP.

1.1 Host Machine Requirements

The following hardware and the software specifications were used during the project. Equivalent or higher specification may also be used.

Category	Specification
Processor	Intel i5 or i7 or AMD Ryzen equivalent
RAM	16 GB recommended and minimum 8 GB
Storage	Minimum 30 GB available
Virtualization	VT-x or AMD-V enabled in BIOS
Operating system	Windows 11

Table 1: Depicts the requirement for host machine.

1.2 VirtualBox Installation

It is used to run the Ubuntu Virtual machine where the Docker WAVSEP and the DAST tools will be installed and executed. Installation steps are as follow:

1. Download the VirtualBox latest version from official website : https://www.virtualbox.org/wiki/Downloads
2. Run the installer and choose default settings.
3. Download and install the VirtualBox extension pack for full functionality.

1.3 Ubuntu Virtual Machine Setup

A dedicated Ubuntu VM is required to host a Docker, Reinforced WAVSEP and the evaluation scripts with all DAST tools.

Settings	Value
VM Name	Ubuntu-DAST
OS Type	Linux
Version	Ubuntu 64 bit
Base Memory	4096 MB recommended (8192 MB if available)
CPUs	2 (recommended : 4)
Hard Disk	40 GB (VDI, Dynamically allocated
Network Adapter	NAT (default)

Table 2: Settings to be done got VM

The following are the steps to create VM:

1. Download Ubuntu 22.04 LTS ISO from : https://ubuntu.com/download/desktop
2. Open the VirtualBox and click “NEW” and then fill the details.
3. Attach the ISO as startup disk and begin the installation.
4. Install the ubuntu using default options.

1.4 Networking Configurations

The Reinforced Benchmark and DAST tools run inside the VM. No special network routing is required for the experiments.The default network setup is recommended which is ADAPTER-1, MODE- NAT. Its purpose is to provide the internet access for installing dependencies. Enable Host only adapter if you want to access the WAVSEP from your host browser. (this is optional.)

1.5 Install Essential Ubuntu packages

Run the following commands inside your VM. This will install all core dependencies that are required for docker, python scripts and tool integration.

- *sudo apt update && sudo apt upgrade -y*
- *sudo apt install curl wget git unzip python3 python3-pip -y*

2 Software Installation

This section details the installation of all the required software components that are inside the Ubuntu virtual machine that includes DAST tools, Benchmark and Python dependencies.

2.1 Install Docker and Docker Compose

It is required to deploy the Reinforced WAVSEP benchmark environment: The commands are as follow:

- *sudo apt update*
  sudo apt install ca-certificates curl gnupg lsb-release -y
- *-curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker.gpg*
- *-echo *

*"deb [arch=$(dpkg --print-architecture) signed by=/usr/share/keyrings/docker.gpg] *
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |

- *sudo tee /etc/apt/sources.list.d/docker.list > /dev/null*
- *sudo apt update*
- *sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin -y*

Verify the installation:

- *docker –version*

docker compose version

2.2 Install Git (Required for cloning Reinforced WAVSEP)

- Command:

- *sudo apt install git -y*

Now, clone the Reinforced WAVSEP Repository. The command:

- *git clone https://github.com/luigiurbano/Reinforced-Wavsep.git*

cd Reinforced-Wavsep

2.3 Deployment of Reinforced Benchmark using docker

The command:

- *sudo docker compose up -d*

The URL for the benchmark: http://localhost:18080/wavsep/

Check for running containers: Command:

- *sudo docker ps*

2.4 Install the python and Required Libraries

Although python3 is pre-installed in Ubuntu, but the pip must be installed manually.
Install pip:

- *sudo apt install python3-pip -y*

Install required python packages:

- *pip3 install pandas numpy lxml*

2.5 Install OWASP ZAP (CLI and GUI support)

Add the ZAP repository and install:

- *sudo add-apt-repository ppa:owasp-zap/ppa*
- *sudo apt update*
- *sudo apt install zaproxy -y*

and then run the ZAP GUI mode: → *zaproxy &*

2.6 Install Arachni Server

Arachni tool is no longer maintained but is still available through archived builds. Download and install through these commands:

- *Wget https://github.com/Arachni/arachni/releases/download/v1.5.1/arachni-1.5.1-0.5.12-linux-x86_64.tar.gz*

tar -xf arachni-1.5.1-0.5.12-linux-x86_64.tar.gz

- *sudo mv arachni-1.5.1-0.5.12 /opt/arachni*

and then add it to the path:

- *echo 'export PATH=$PATH:/opt/arachni/bin' >> ~/.bashrc*

source ~/.bashrc

and run it:

- *arachni http://localhost:18080/wavsep/ --output-format=json --report-save-path=arachni_output*

2.7 Install Wapiti Scanner

Wapiti is available through apt. Installation command:

- *sudo apt install wapiti -y*

Run the tool :

- *wapiti http://localhost:18080/wavsep/ -f json -o wapiti_wavsep.json*

2.8 Install Java which is required for ZAP and some other tools

- *sudo apt install default-jre -y*

2.9 Install Additional Utility Tools

TOOL	Purpose	Command
unzip	Extract ZIP files	*Sudo apt install unzip -y*
htop	Performance monitoring	*Sudo apt install htop -y*
net-tools	Network diagnostics	*Sudo apt install net-tools -y*
curl	URL testing	*Sudo apt install curl -y*

Table 3: Additional utility tools with purpose and command to install

3 Test Scanners (DAST Tool Execution Procedures)

This section describes how to execute the three Dynamic Application Security Testing (DAST) tools used in this project—OWASP ZAP, Arachni and Wapiti—against the Reinforced WAVSEP benchmark. Each scanner is run in a controlled, black-box configuration, using default settings to reflect practical real-world usage.

3.1 Overview

- All scanners are executed **externally** to the WAVSEP Docker container. Each tool interacts with the benchmark exclusively via HTTP at: http://localhost:18080/wavsep/
- No authentication, proxy rewriting, or tuned payloads were used. The outputs from each scanner are stored inside the /scans/ directory:

Scanner	Output Format	Output File
OWASP ZAP	XML	Zap_wavsep_Results.xml
Arachni	JSON	Arachni_full_scan_no_dom.xml
Wapiti	JSON	Wapiti_wavsep.json

Table 4: Different scanners' output file and format

3.2 Executing OWASP ZAP Scan

3.2.1 Executing OWASP ZAP Scan

- ZAP 2.x installed on Ubuntu VM
- Java Runtime Environment
- Internet disabled (local-only scanning)

3.2.2 ZAP Launch Modes

ZAP may be executed either via GUI or command line.
For reproducibility, **command-line active scanning** is recommended.
Command:

- *zap.sh -cmd -quickurl http://localhost:18080/wavsep/ \ -quickout ZAP_WAVSEP_Results.xml*

3.2.3 Configuration Applied

Feature	Setting
Scan Mode	Active Scan
Spider	Enabled (Default depth)
Scope	Restricted to WAVSEP root
Authentication	None
DOM XSS	Enabled by default
Reporting	XML export

Table 5: Different configurations and their settings

- After completion: *DAST_Evaluation/scans/ZAP_WAVSEP_Results.xml*

3.3 Executing Arachni Scan

3.3.1 Requirements

- Arachni Framework 1.6 installed
- JSON report output enabled

3.3.2 Scan Command

Arachni is executed in **full-scan** mode with DOM analysis disabled (consistent with your environment).
Command:

- *arachni http://localhost:18080/wavsep/ \ --output-only-positives=false \ --report-save-path=arachni_wavsep.afr*

Convert to JSON :

- *arachni_reporter arachni_wavsep.afr --reporter=json:outfile=arachni_full_scan_no_dom.json*

3.3.3 Configuration Applied

Parameter	Value
Scan Type	Full scan
DOM Inspector	Disabled
Payload Set	Default
Plugins	Default
Max depth	Unlimited (default by crawler
Report	JSON

Table 6: Different parameters and values to set

Expected Output: → *DAST_Evaluation/scans/arachni_full_scan_no_dom.json*

3.4 Executing the Wapiti Scan

3.4.1 Requirements

- Wapiti 3.x installed
- Python3 installed

3.4.2 Scan Command

➢ *wapiti http://localhost:18080/wavsep/ *
-f json -o wapiti_wavsep

This generates:

- wapiti_wavsep.json
- Additional auxiliary log files

3.4.3 Configuration Applied

Parameter	Value
Format	JSON
Injection	Default enabled
Crawl Depth	Default Recursive
Authentication	None
Timeout	Default
Report	JSON

Table 7: Different parameters and the setting value

➢ Output Location: *DAST_Evaluation/scans/wapiti_wavsep.json*

4 Code Setup

This section helps to describes how to configure, organise and execute the automated evaluation framework that is used to benchmark the OWASP ZAP, Arachni and Wapiti against the Reinforced WAVSEP dataset. It includes the directory structure, required scripts and dependencies.

4.1 Directory Structure

All the evaluation scripts and the scans output should be stored in a single working directory for consistency: The recommended directory layout should look like this:

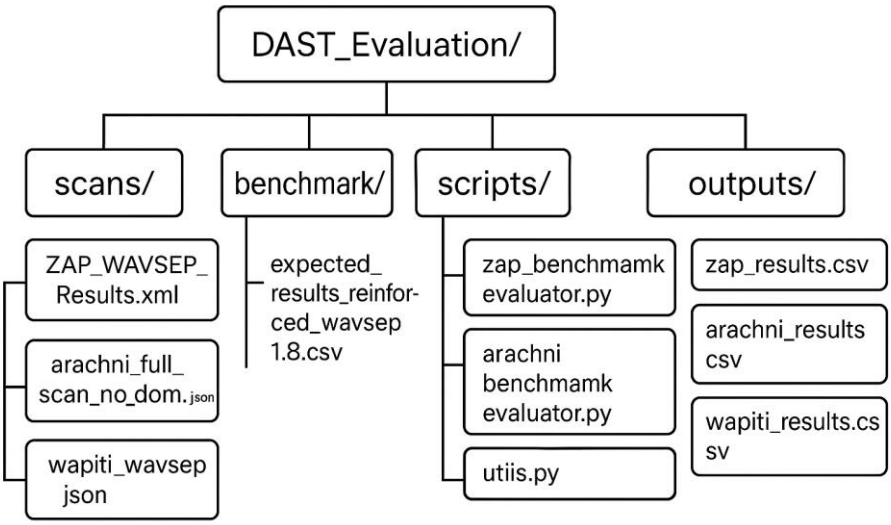

Figure 1: Recommended directory structure

Folder	Description
/scans	Contains raw XML/JSON scan outputs from three tools.
/benchmark	Contains Reinforced WAVSEP ground truth file CSV
/scripts	Python evaluation scripts
/outputs	Final exported CSV performance metrics

Table 8: Different folders and their use

4.2 Required Python files

Every tool uses a dedicated Python evaluation script and one shared utility module. They can be found in artefacts:

The script summary looks like:

Script name	Purpose
Zap_benchmark_evaluator.py	Parses ZAP xml, maps detections and compute metrics
Arachni_benchmark_evaluator.py	Processes Arachni JSON output
Wapiti_benchmark_evaluator.py	Processes Wapiti JSON output
Utils.py	Shared helper function for normalization and mapping metrics.

Table 9: Different python scripts and their use

4.3 Create the Evaluation script folder

Command:

- *mkdir -p DAST_Evaluation/{scans,benchmark,scripts,outputs}*

cd DAST_Evaluation

4.4 Placement of required files

Move the scanner outputs:

- *cp ~/WAVSEP_Artefact/scan_results/ZAP/ZAP_WAVSEP_Results.xml scans/*
- *cp~/WAVSEP_Artefact/scan_results/Arachni/arachni_full_scan_no_dom.json scans/*
- *cp ~/WAVSEP_Artefact/scan_results/Wapiti/wapiti_wavsep.json scans/*

Also, copy the benchmark ground truth:

- *cp ~/Reinforced-Wavsep/expected_results_reinforced_wavsep-1.8.csv benchmark/*

4.5 Running the ZAP evaluation script

Move into the scripts folder:

- *cd DAST_Evaluation/scripts*

Run the evaluator:

- *python3 zap_benchmark_evaluator.py*

Expected output Files: They are generated under /outputs/ :

File	Description
Zap_results.csv	TP/FP/FN and metrics per category
Zap_detections.csv	Raw mapped detections
Zap_summary.txt	Overall precision, recall and F1

Table 10: Output files and their description

```
=== Benchmark Scores (ZAP vs Reinforced-WAVSEP) ===
Total WAVSEP test cases: 1429
Total with a ZAP detection (by test_name): 227

TP: 539
FP: 40
FN: 850
TN: 0
Precision: 0.931
Recall: 0.388
F1: 0.548

=== Per-category metrics ===
[cmdi] tests=121 TP=117 FP=0 FN=4 TN=0 P=1.000 R=0.967 F1=0.983
[pathtraver] tests=941 TP=110 FP=14 FN=817 TN=0 P=0.887 R=0.119 F1=0.209
[redirect] tests=69 TP=60 FP=9 FN=0 TN=0 P=0.870 R=1.000 F1=0.930
[sqli] tests=174 TP=135 FP=10 FN=29 TN=0 P=0.931 R=0.823 F1=0.874
[xss] tests=124 TP=117 FP=7 FN=0 TN=0 P=0.944 R=1.000 F1=0.971
```

Figure 2: Shows the result after running the ZAP evaluation script

4.6 Running the Arachni Evaluator Script

- *python3 arachni_benchmark_evaluator.py*

Expected outputs:

File	Description
Arachni_results.csv	Category metrics
Arachni_summary.txt	Overall evaluation summary

```
=== Benchmark Scores (Arachni vs Reinforced-WAVSEP) ===
TP: 1102
FP: 22
FN: 287
TN: 18
Precision: 0.980
Recall: 0.793
F1: 0.877

=== Per-category metrics ===
[cmdi] tests=121 TP=113 FP=0 FN=8 TN=0 P=1.000 R=0.934 F1=0.966
[pathtraver] tests=941 TP=927 FP=10 FN=0 TN=4 P=0.989 R=1.000 F1=0.995
[redirect] tests=69 TP=60 FP=7 FN=0 TN=2 P=0.896 R=1.000 F1=0.945
[sqli] tests=174 TP=0 FP=4 FN=164 TN=6 P=0.000 R=0.000 F1=0.000
[xss] tests=124 TP=2 FP=1 FN=115 TN=6 P=0.667 R=0.017 F1=0.033

Saved detailed evaluation CSV to: ARACHNI_WAVSEP_Evaluation_Output.csv
```

Figure 3: Shows the result after running the Arachni evaluation script

4.7 Running the Wapiti scanner

- *python3 wapiti_benchmark_evaluator.py*

Expected outputs:

File	Description
Wapiti_results.csv	Category metrics
Wapiti_summary.txt	Overall evaluation summary

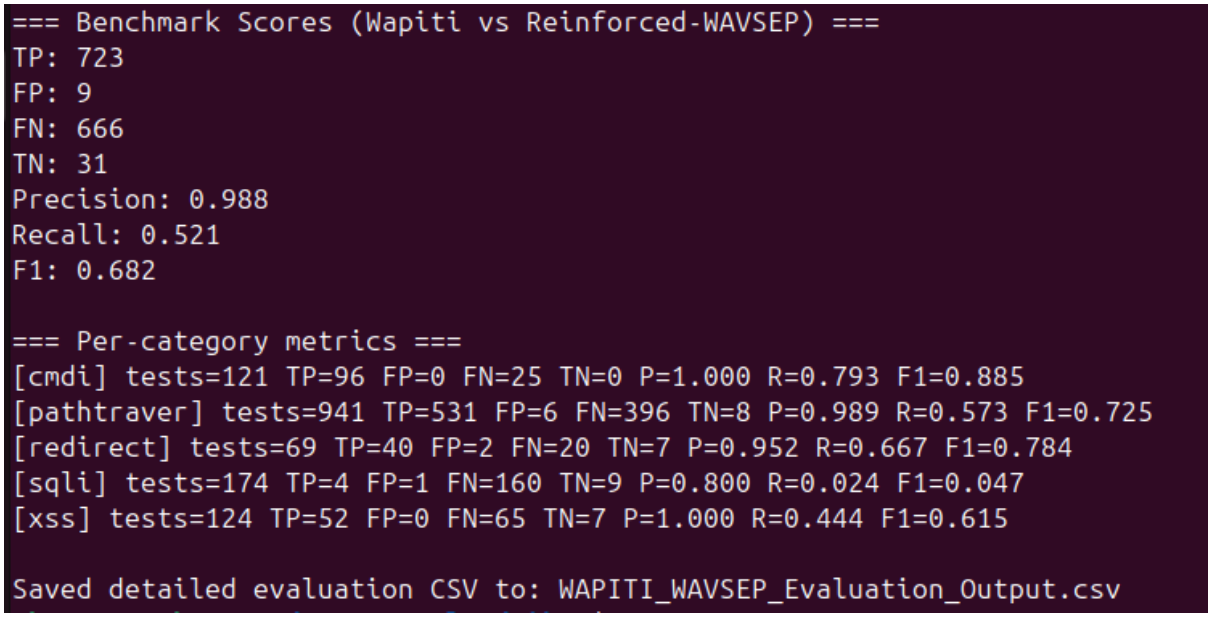

Figure 4: Shows the result after running the Wapiti evaluation script

References

Docker Inc. (2024) *Docker Engine Overview.* Available at: https://docs.docker.com/engine/

Docker Inc. (2024) *Docker Compose Documentation.* Available at: https://docs.docker.com/compose/

Oracle Corporation (2024) *VirtualBox User Manual*. Available at: https://www.virtualbox.org/manual/

Canonical Ltd. (2024) *Ubuntu 22.04 LTS Documentation*. Available at: https://ubuntu.com/server/docs

OWASP Foundation (2024) *OWASP Zed Attack Proxy (ZAP) Documentation*. Available at: https://www.zaproxy.org/docs/

Arachni Project (2020) *Arachni Web Application Security Scanner Framework*. Available at: https://github.com/Arachni/arachni

IFRI-CIRCL (2024) *Wapiti 3 Web Vulnerability Scanner*. Available at: https://github.com/wapiti-scanner/wapiti

Python Software Foundation (2024) *Python 3 Documentation*. Available at: https://docs.python.org/3/

The Pandas Development Team (2024) *Pandas 2.x Documentation*. Available at: https://pandas.pydata.org/docs/

Urbano, L. (2023) *Reinforced WAVSEP Benchmark.* Available at: https://github.com/andresriancho/wavsep-reinforced